FPGA Design
EECE 277

Reconfigurable Computing

Dr. William H. Robinson
April 18, 2005

http://eecs.vanderbilt.edu/courses/eece277/

Topics
• Administrative stuff
  – Return graded Exam 2
  – Demo final project on or before April 26 (return UP2 kits)
  – Final project report and presentation due April 30 (exam block)
  – Questions on final project
• Reconfigurable computing defined
• Applications

Final Project
• 20% - Project Proposal
• 25% - Demonstration
• 35% - Final Project Report
• 10% - Final Project Presentation
• 10% - Team Member Evaluation

Processor Efficiency
• Computational Density Trend

Adapted from John Wawrzynek’s CS 294-3 lecture notes. Copyright © 2004 UCB.
Challenging Our Assumptions

“Are we making copies in sub-micron CMOS VLSI of copies in NMOS of copies in TTL of early vacuum tube computer designs?”

– Andre DeHon
California Institute of Technology

- 10000x increase in single-chip silicon capacity changes the underlying design costs
  - Von Neumann architectures designed to heavily time-multiplex the expensive ALU resource

- General-purpose computing machines don’t have to look like processors.

What and Why?

- What is “reconfigurable computing (RC)?”
- Many definitions

- Our “standard” definition:
  Computing via a post-fabrication and spatially programmed connection of processing elements.
  - FPGA implementation of a processor core to run a program is excluded - not spatial mapping of problem
  - ASIC implementations excluded – not post-fabrication programmable

- Does this include arrays of processors?
- Often the definition restricts RC to mapping to “fine-grained” devices (such as FPGAs)

Spatial Computation

- Example:
  grade = 0.2 x PP + 0.25 x D + 0.35 x FPR + 0.1 x FPP + 0.1 TME

- A hardware resource (multiplier or adder)
  - Allocated for each operator in the compute graph

- The abstract computation graph becomes the implementation template

Temporal Computation

- A hardware resource is time-multiplexed to implement the actions of the operators in the compute graph

- Close to a sequential processor/software solution

- Many in-between cases exist.
An Important Distinction

- Instruction binding time
  - When do we decide what operation needs to be performed?

- General principle
  - Earlier the decision is bound, the less area and delay required for the implementation

RC Strategy

- Exploit cases where operation can be bound and then reused a large number of times

- Customize for operator type, width, and interconnect

- Low-overhead exploitation of application parallelism

Advantages of RC

- Conventional processors have three major sources of inefficiency:
  - Heavy time multiplexing of Function Units (ALUs)
  - Instruction issue overhead
  - Memory hierarchy to deal with memory latency
Advantages of RC

- **Relative to microprocessors**
  - On average, higher percentage of peak (or raw) computational density is achieved with reconfigurable devices.

- **Fine-grain flexibility leads to exploitation of problem specific parallelism at many levels**

- **Also, many different computation models (or patterns) can be supported**
  - In general, it is possible to match problem characteristics to hardware, through the use of problem specific architectures and low-level circuit specialization.

Spatial mapping of computation versus multiplexing of function units (as in processors)

- Relieves pressure for memory capacity, BW.
- Low-latency and local communication patterns.

Modern FPGAs make good system-level components:

- Relatively large number of IOs (many parallel memory ports)
- High-BW communications
- Machines based on these components can easily scale peak performance by riding Moore's curve (FPGAs are process drivers)
- Low-level redundancy permits fault-tolerance and great cost savings
- Built-in microprocessors

Even in an application with fixed algorithms, reconfigurable devices may offer advantages over a full-custom or ASIC approach:

- FPGAs are processes drivers, therefore a generation ahead of ASIC.
- Increasing Non-Recurring Engineering costs (NREs) for ASIC and full-custom has pushed "cross-over" point way out.
- Time to market advantage.
- Programmability leads to:
  - Project risk management
  - Extended product life-times.
FPGAs vs. ASIC Cost-Argument

- **ASIC**: High NRE costs ($2M for 0.35um chip). Relatively Low cost per die.
- **FPGAs**: Very low NRE costs. Relatively low silicon efficiency yields high cost per part.
- **Cross-over volume** from cost effective FPGA design to ASIC in the 10K range.

Cross-Over Point is Moving Right

- **ASIC**: Increasing NRE costs ($40M for 90nm chip1) (mask costs2, verification, etc.)
- Limited number of standard silicon designs becomes inevitable.
- **FPGAs**: Obvious candidate for one of those designs, furthermore, FPGAs better able to follow Moore’s Law, relatively cheaper to test.

1 Vahid Manian, VP manufacturing and operations, Broadcom Corp.
2 Roger Minear, Agere Systems Inc, 30 to 35-layer mask set approx. $650,000 for 130nm and $1.4M for 90nm.

Post-Fabrication Customization

- **Gate Array** like devices return to fill the gap
  - Post-fab customization with limited mask layers or direct-write ebeam
  - LSI Logic Rapid-Chip, CMU/VPGA, e-beam programmable devices
  - Lower NREs than ASICs, more silicon efficiency than FPGAs
- **So, why bother with FPGAs?**

FPGAs are Reconfigurable

Seemingly obvious point but …

1. **Commercial applications have not taken advantage of reconfigurability**
   - Xilinx/Altera haven’t done much to help
   - Methodologies/tools nearly nonexistent
2. **Volume/cost graphs don’t accurately capture the potential real costs and other advantages**

Reconfiguration uses:

- Field upgrades yields product life extension, changing requirements.
- In system board-level testing and field diagnostics.
- Tolerance to manufacturing faults.
- Risk-management in system development.
- **Runtime reconfiguration** yields higher silicon efficiency.
  - Time-multiplexed pre-designed circuits take maximum use of resources.
  - Runtime specialized circuit generation.
Advantages of RC

• **Dynamic reconfiguration** might permit even higher efficiency through hardware sharing (multiplexing) and on the fly circuit specialization.
  – Largely unexploited (unproven) to date
  – A few research projects have explored this idea

Dynamic Reconfiguration

• **Why dynamic reconfiguration?**

1. Time-multiplexing resources allows more efficient use of silicon (in ways ASICs typically do not):
   a. Low-duty cycle or "off critical path" computations time share fabric while critical path stays mapped in:

2. **Fabric virtualization** allows automatic migration up and down in device sizes and eases application development.

3. Runtime Circuit Specialization:
   • Example: fixed coefficient multipliers in adaptive filter changing value at low rate.
   • Aggressive constant propagation (based perhaps on runtime profiling), reduces circuit size and delay.
   • Could use “branch/value/range prediction” to map most common case and fault in exceptional cases.
Multi-Modal Computing Tasks

The premier applications for reconfigurable devices are those with constrained size/weight, need multiple functions at near ASIC performance.

- **Mini/Micro-UAVs**
  - One piece of silicon for all of sensor processing, navigation, communications, planning, logging, etc.
  - At different times different tasks take priority and consume higher percentage of resources.
- Other examples: hand-held multi-function device with GPS, smart image capture/analysis, communications.

Multiple ASICs too expensive/big. Processor too slow. Fine-grained reconfigurable devices has the flexibility to efficiently match task parallelism over a wide variety of tasks – deployed as needed.

---

Garp – Hybrid Processor Model

- Pre-generated circuits for common program kernels cached within reconfigurable array and used to accelerate MIPS programs.
- nsec configuration swap time
- Limited speedup – tied to single execution thread

![Garp Diagram]


<table>
<thead>
<tr>
<th>Function</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>sifon (76)</td>
<td>1.77</td>
</tr>
<tr>
<td>sifon (1024)</td>
<td>14</td>
</tr>
<tr>
<td>sort</td>
<td>2.1</td>
</tr>
<tr>
<td>image median filter</td>
<td>28.1</td>
</tr>
<tr>
<td>DES (ECB mode)</td>
<td>19.6</td>
</tr>
<tr>
<td>image filtering</td>
<td>18.3</td>
</tr>
</tbody>
</table>

Speedups over 4-way superscalar UltraSparc on same process and comparable die size and memory system.